**Отчет по ИДЗ №1**

**По предмету “Операционные системы”**

**Морина Ильи Олеговича**

**БПИ215**

**Вариант 25**

**Тема «Использование каналов»**

**Задание:** Разработать программу, которая определяет в ASCII-строке частоту встречаемости различных идентификаторов, являющихся словами, состоящими из букв и цифр, начинающихся с буквы. Разделителями являются все другие символы. Для тестирования можно использовать программы, написанные на различных языках программирования.

**Как работает** мой алгоритм (функция для решения задачи). Так как техническое задание не совсем четкое, то на усмотрение исполнителя (в лице меня) было предпринято следующее: предположим на вход поступает строка “435HGH;HLJGG675;;JHGFH000//76j” в таком случае моя программа выдаст ответ 2 (то есть идентификатора 2). В данном примере идентификаторами являются HLJGG675 и HGFH000 так как они во-первых начинаются с буквы, во-вторых находятся между разделителей. 435HGH и 76j не являются идентификаторами так как начинаются с цифр.

Примечание: некоторые части кода и логика взята из презентаций и программ к семинарам.

**Содержимое ИДЗ:**

Три папки с заданием на каждую оценку

IDZ1\_1 на 4

IDZ1\_2 на 5

IDZ1\_3 на 6

В каждой папке будет папка tests (везде они одинаковые). Так же помимо файлов с входными данными (тестами) там есть файлы с примерами выходных данных (но IDZ1\_2 немного другой формат вывода, поэтому стоит смотреть output файлы), т е test.out файлы это то что должно выводиться на соответствующих тестах.

Помимо тестов будет соответственно файл с кодом, в котором будут **комментарии.**

Так же 5 output файлов с результатами работы соответствующих тестов.

**Тестовое покрытие** для всех заданий одинаковое и состоит из следующих 5 тестов:

**Тест 1**

RTRT123;;IUYIOY8876//9876JK|JJH

Обычная строка из идентификаторов с разделителями - результат 3.

**Тест 2**

1324……HHH…//..8877KKK;;LLKJHG9

Начинается с цифр, много разделителей - результат 2.

**Тест 3**

Kjjhgfghj1234>>>234543>>>0000jjhkj>-009873H;JHHHHH

Есть маленькие латинские буквы, использован новый разделитель «>» - результат 2.

**Тест 4**

Lkjhgfghjk/////////////////098765456789////kjhgfdfghj098765’’’’’’’’’’’’’’456hgf’’’’lkjhgfdfghj0987654’’’kjhgfghjlkj098;;;kjhgcf45678;;;,mnbv888……sdfghjkjhgfd345678jhgfd||||||||||||||||||234543df

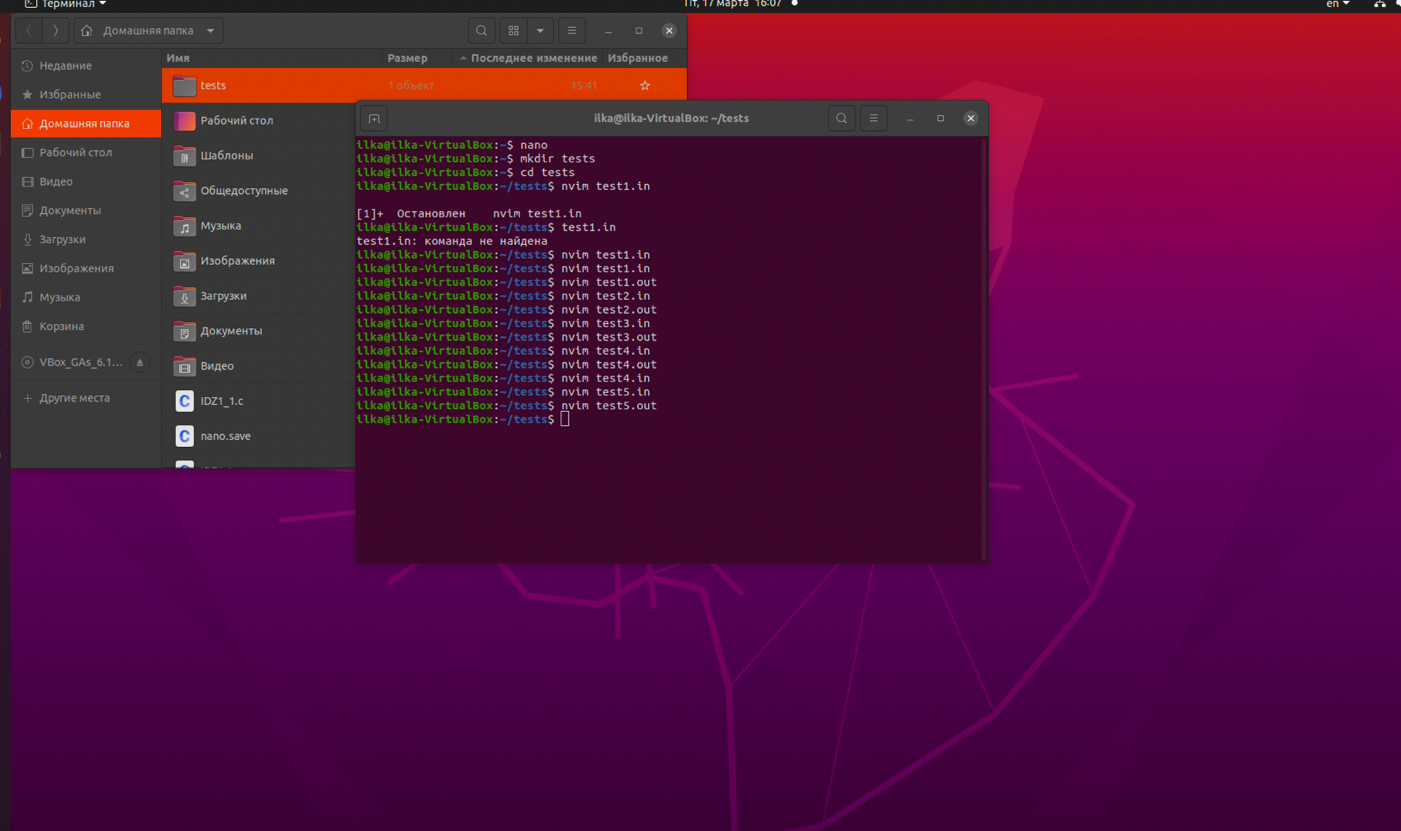
Длинная строка - результат работы - 7.

Для каждой программы будет отдельно сформирована папка с тестами (одинаковыми) и с выходными файлами после отработки на каждом тесте.

**Тест 5**

98765467sdfgdfb;;;09876ghjk;;;76543dfgh;;7vbnm////6543dfgh

Строка только из «идентификаторов», которые начинаются только с цифр, то есть не являются идентификаторами, соответсвенно результат – 0.



**Замечание по тестированию.**

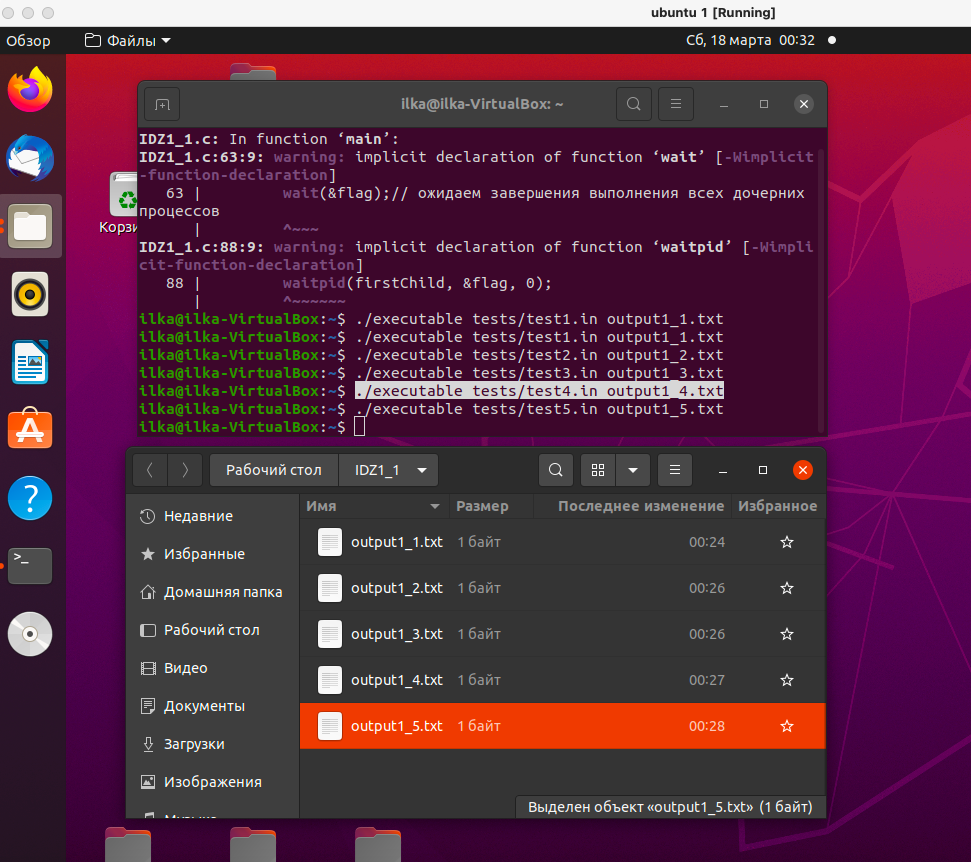
Стоит отметить, что при множеством тестирование может произойти ситуация, когда в выходной файл записывается содержимое входного вместо результата, который должен быть, такое может произойти из-за того, родительский поток отрабатывает быстрее и закрывает остальные, ну либо же любая другая ситуация. В таком случае возможно лучше было бы использовать вместо функции write() функцию fprintf(), которая стабильнее работала с буфером и потоками, но по заданию надо было использовать write() и read(), что и было сделано.

**Решение на отметку 4:**

**Текстовая схема решения.**

1. Родительский процесс создает два дочерних процесса - Дочерний 1 и Дочерний 2, используя системный вызов **fork()**.
2. Дочерний 1 процесс читает входной файл и записывает его содержимое в канал, используя системные вызовы **open()**, **read()** и **write()**.
3. Дочерний 2 процесс ожидает завершения Дочернего 1 процесса, используя системный вызов **waitpid()**.
4. Дочерний 2 процесс читает содержимое канала, в котором были записаны данные Дочерним 1 процессом.
5. Дочерний 2 процесс анализирует содержимое канала и находит количество идентификаторов, записывая результат в строку.
6. Дочерний 2 процесс записывает результат анализа в другой канал.
7. Дочерний 3 процесс ожидает завершения Дочернего 2 процесса, используя системный вызов **waitpid()**.
8. Дочерний 3 процесс читает содержимое канала, в котором были записаны данные Дочерним 2 процессом.
9. Дочерний 3 процесс записывает результат анализа в выходной файл, используя системные вызовы **open()** и **write()**.
10. Родительский процесс ожидает завершения Дочернего 2 процесса, используя системный вызов **wait()**.

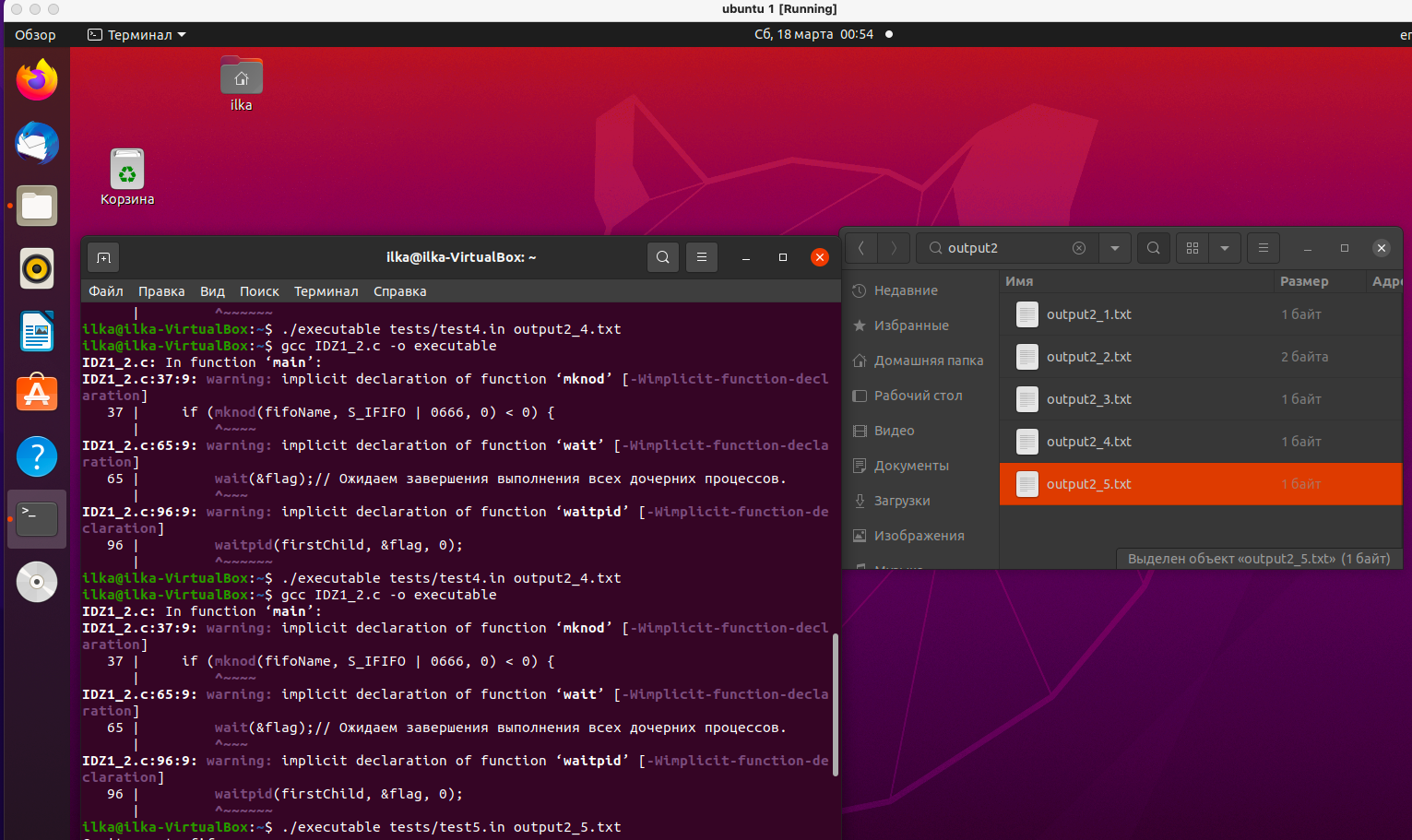
Провели тесты для данной программы и поместили их в отдельную папку. Если посмотреть содержимое выходных файлов, то видно, что все работает корректно.



**Решение на отметку 5:**

**Схема работы программы:**

1. Создание именованного канала с помощью функции mknod.
2. Проверка корректности количества аргументов командной строки.
3. Создание двух дочерних процессов с помощью функции fork.
4. Получение имен входного и выходного файлов из аргументов командной строки.
5. Родительский процесс ожидает завершения выполнения всех дочерних процессов.
6. Первый дочерний процесс (FirstChild) открывает входной файл, считывает из него данные, записывает их в именованный канал с помощью функции write, и закрывает файл.
7. Второй дочерний процесс (SecondChild) открывает именованный канал на чтение, считывает данные из канала с помощью функции read, подсчитывает количество идентификаторов с помощью функции findIdentifier, открывает выходной файл, записывает результат в файл с помощью функции write, и закрывает файл.
8. Каждый дочерний процесс завершается после выполнения своей работы.



**Замечание**

﻿if (mknod(fifoName, S\_IFIFO | 0666, 0) < 0 && errno != EEXIST) {

perror("mknod");

exit(EXIT\_FAILURE);

}

Для удобства тестирования была отключена проверка на существование fifo файла с заданным именем, то есть программу можно запускать множество раз и не придется менять имя fifo файла в коде, но в реальной жизни это может вызвать ряд проблем, поэтому каждый раз надо создавать уникальный fifo файл.

**Решение на отметку 6**

Первый канал (firstToSecond) используется для передачи данных из первого дочернего процесса во второй дочерний процесс. Второй канал (secondToFirst) используется для передачи результата работы второго дочернего процесса в первый дочерний процесс.

Программа принимает два аргумента: имя входного файла и имя выходного файла. Если аргументов не два, то программа выводит сообщение об ошибке и завершается.

Первый дочерний процесс открывает входной файл, считывает данные в буфер и записывает их в первый канал. Затем закрывает дескриптор записи в первый канал и ожидает результата от второго дочернего процесса во втором канале. Получив результат, он записывает его в выходной файл.

Второй дочерний процесс считывает данные из первого канала, ищет и подсчитывает количество идентификаторов, записывает результат во второй канал и завершается.

Родительский процесс создает первый дочерний процесс и ждет его завершения. Если первый дочерний процесс завершается успешно, то он создает второй дочерний процесс и ждет его завершения. Если второй дочерний процесс завершается успешно, то родительский процесс завершается.

Тесты так же проведены, результаты успешны.

